Modernizing Scientific Programming

Charles D. Norton

National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109-8099

High Performance Computing Systems and Applications Group
http://www-hpc.jpl.nasa.gov/PEP/nortonc

Collaborators

Viktor Decyk

Deapartment of Physics and Astronomy
University of California at Los Angeles, Los Angeles California

Dinshaw Balsara

National Center for Supercomputing Applications
University of Illinois at Urbana-Champagne

Gunnar Ledfelt

Department of Numerical Analysis and Computing Science
Uppsala University and Royal Institute of Technology, Sweden

John Lou

NASA Jet Propulsion Laboratory
California Institute of Technology, Pasadena,California

Boleslaw Szymanski

Department of Computer Science
Rensselaer Polytechnic Institute, Troy, New York

Why Fortran 90?

Motivation

- Ambitious parallel scientific computations impose new demands on software development

Approach

- Study of object-oriented concepts and their application to scientific computing with Fortran 90

Significance

- Modernizes scientific programming without sacrificing performance

Parallel Computing

Importance

- Supports solution of very large problems, leading to better science

Issues

- Requires sophisticated programming to achieve high performance
- Represents the best alternative to make significant new advances in many fields of science and engineering

The Numerical Tokamak Turbulence Project

Participating Institutions

LLNL, LANL, UCLA, PPPL, NERSC, ORNL, UTA, GA, UColorado

Tokamak Definition

Torodial fusion energy device using magnetic confinement

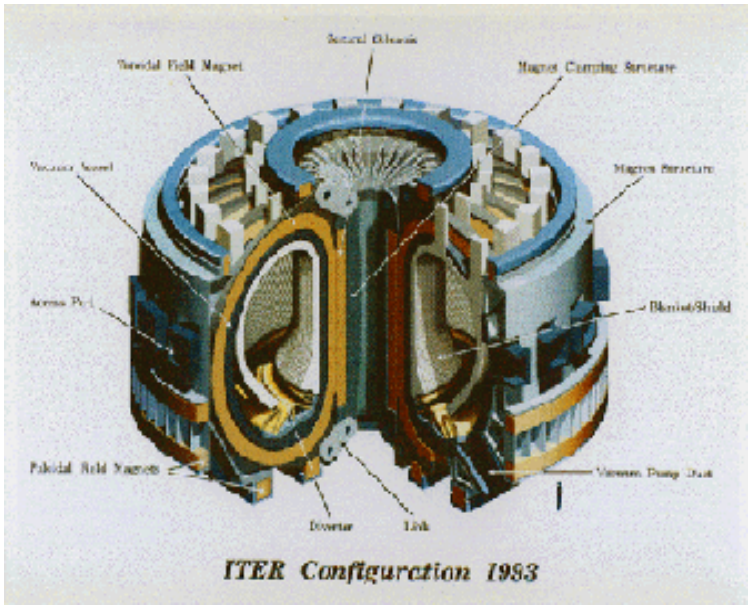
Physics & Computational Goal

Model transport of particles & energy in tokamaks

Demonstrate MPP usability

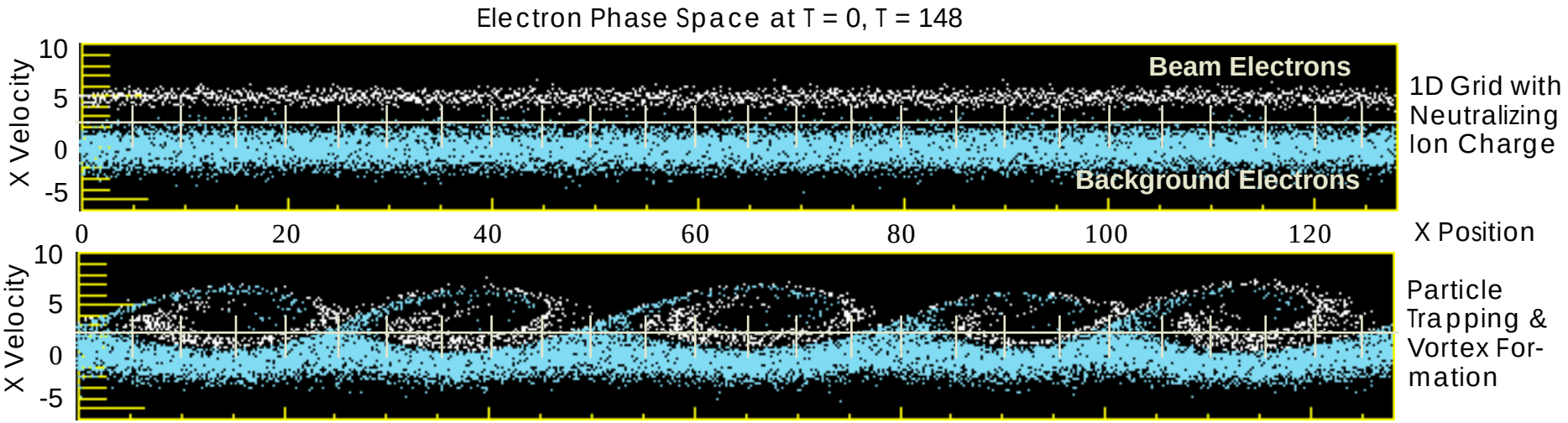
Competing Approaches

Gyrofluid: Enhanced MHD
Gyrokinetic: Reduced PIC



ITER Configuration 1993

Earth and Space Sciences Project (625-20)



Beam-Plasma Instability Experiment Which Drives Plasma Waves To Saturation

The Computation Cycle

- Initialize particle positions
- Interpolate charge density to grid
- Calculate electric field
- Apply force on particles from known E field
- Update particle positions from field

Charge Deposition

$$q(\boldsymbol{x}) = \sum_i q_i \bullet S(\boldsymbol{x} - \boldsymbol{x}_i)$$

Field Solve

$$-\nabla^2\Phi = 4\Pi q(\boldsymbol{x})$$

$$\boldsymbol{E}(\boldsymbol{x}) = -\nabla\Phi$$

Particle Push

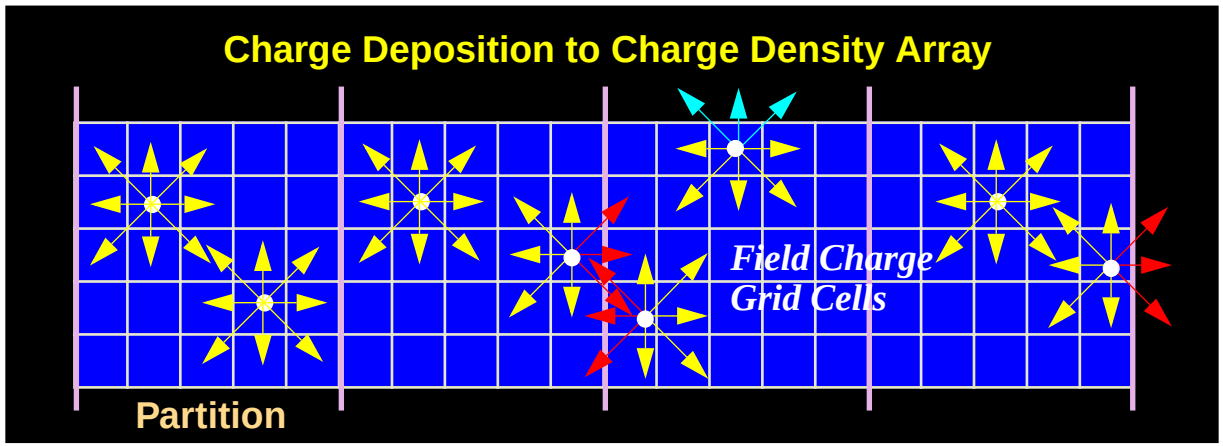
$$\frac{d\boldsymbol{v}_i}{dt} = \frac{q_i}{m_i}(\boldsymbol{E}(\boldsymbol{x}_i)) \qquad \frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{v}_i$$

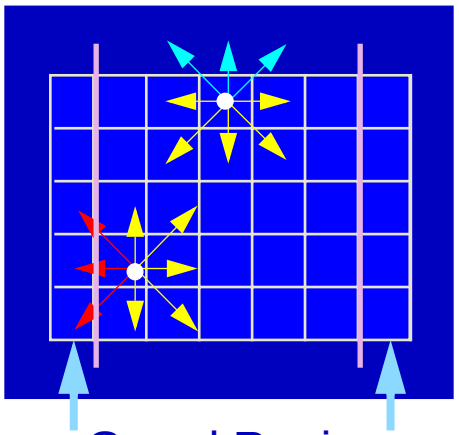
$$\boldsymbol{v}_i\left(t + \frac{\Delta t}{2}\right) = \boldsymbol{v}_i\left(t - \frac{\Delta t}{2}\right) + \frac{F_i(t)}{m_i}\Delta t$$

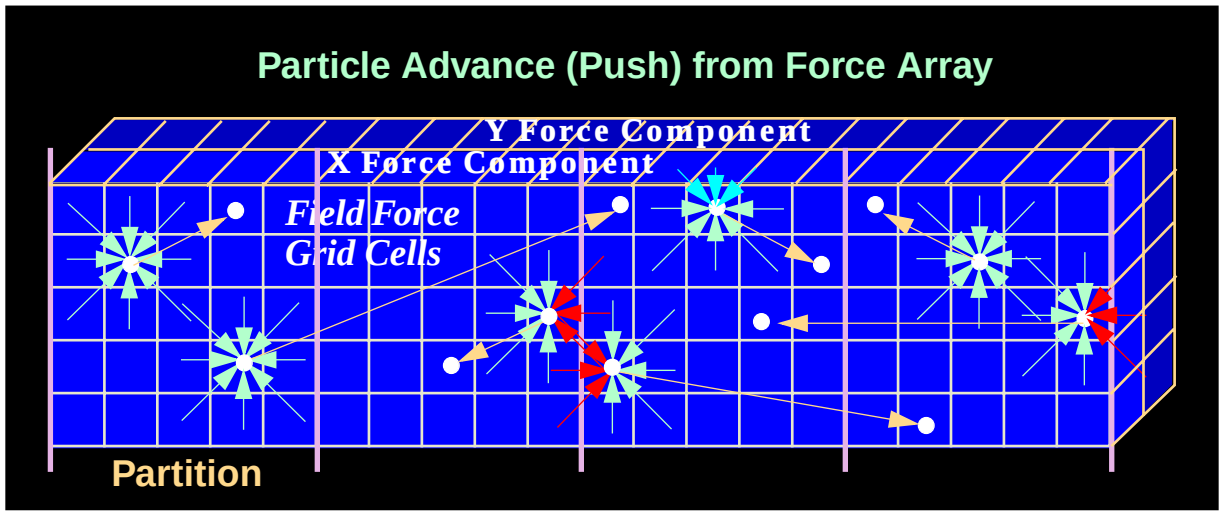
$$\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i\left(t + \frac{\Delta t}{2}\right)\Delta t$$

Plasma PIC Algorithm Particle/Field Interaction in 2-dim

SPMD Data Parallel Model with Explicit Message Passing







Field and particle partitioning may differ based on load balancing requirements...

Object-Oriented Programming Concepts in a Nutshell

CLASSES

- Contain user-defined types and the procedures that work on them

USER-DEFINED TYPES

- Allow one to combine into a single structure related data which can be passed together to procedures. Internal details of the structure can be changed without impacting the clients (users)

INHERITANCE

- Allows a family of similar types to share common code. The family has a special data relationship, where the child adds features to the parent

RUN-TIME POLYMORPHISM

- Allows one to write code for a family of types, where the action taken varies dynamically (at run time) based on the type

Fortran 90 Features Modernize Programming

Modules
Encapsulate data and routines across program units.

Generic Interfaces
A single call can act differently based on the parameters.

Dervied Types
Allow user-defined types to be created.

Use Association
Supports module interaction.

Array Syntax
Simplifies operations on whole arrays, or array components.

Pointers/Allocatable Arrays
Support the use of dynamic data structures.

Fortran 90 is a subset of High Performance Fortran, and is backward compatible with Fortran 77

PIC Example of Fortran 90 Object-Oriented Concepts

Many arguments to describe features (Fortran 77)

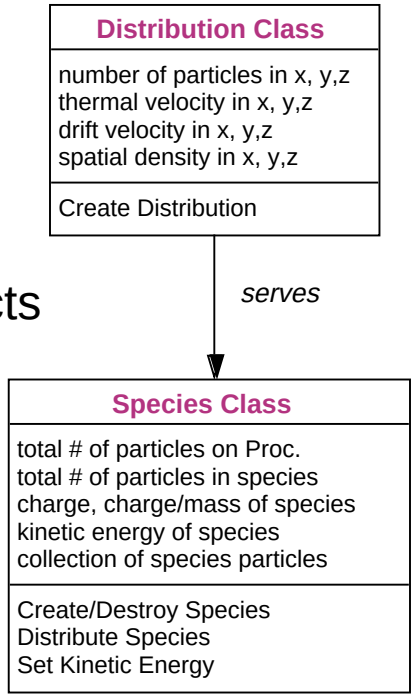
```
dimension part(idimp,npmax), q(nx,ny,nzpmx)
dimension fx(nx,ny,nzpmx),fy(nx,ny,nzpmx),fz(nx,ny,nzpmx)
data qme, dt /-1.,.2/
call push(part,fx,fy,fz,npp,qme,dt,wke,nx,ny,npmax,nzpmx)
call dpost(part,q,npp,noff,qme,nx,ny,npmax,nzpmx)
```

Combine into logically related units (Fortran 90)

```
use partition_module; use plasma_module
type (species) :: electrons
type (sfield) :: charge_density
type (vfield) :: efield
type (slabs) :: edges
real :: dt = .2
call plasma_push(electrons, efield, edges, dt)
call plasma_dpost(electrons, charge_density, edges)
```

Organizing Codes Using Object-Oriented Techniques

- Relationships among data organize the code, rather than relationships among procedures
- Programming consists of iteratively:
  - "Finding" the objects
  - Understanding and building relationships among objects
- Many kinds of relationships can exist...
  - Links represent connections between classes
  - An aggregration is an object assembled from component objects
  - Inheritance represents a family of related abstract data types with similar properties



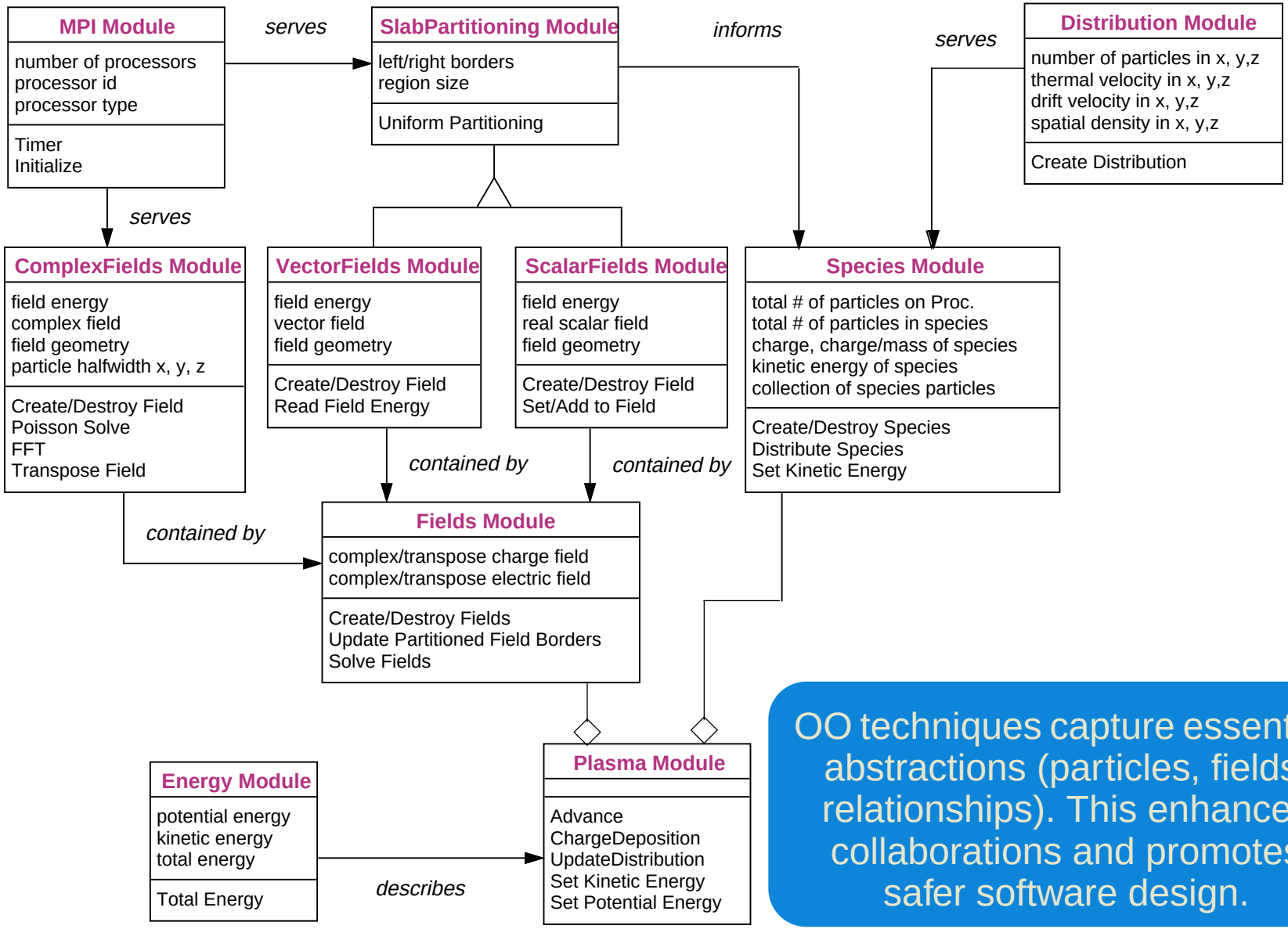
Fortran 90 Modules Provide Abstraction Features

```
MODULE species_class
USE distribution_class; USE slab_partition_class
IMPLICIT NONE
TYPE particle                                    ! derived type...
   PRIVATE
   real :: x,y,z,vx,vy,vz
END TYPE particle
TYPE species                                     ! derived type...
   real :: qm,qbm,ek
   integer :: nop,npp
   TYPE (particle),dimension(:),pointer::p
END TYPE species
CONTAINS
SUBROUTINE spec_dist(this,edges,distf)           ! member fcn...
   TYPE (species), intent(out) :: this           ! the object...
   TYPE (slab), intent(in) :: edges
   TYPE (distfcn), intent(in) :: distf
   ! code omitted...
END SUBROUTINE spec_dist    ! additional member functions...
END MODULE species_class
```

"Sketch" Of Object-Oriented Fortran 90 PIC Code

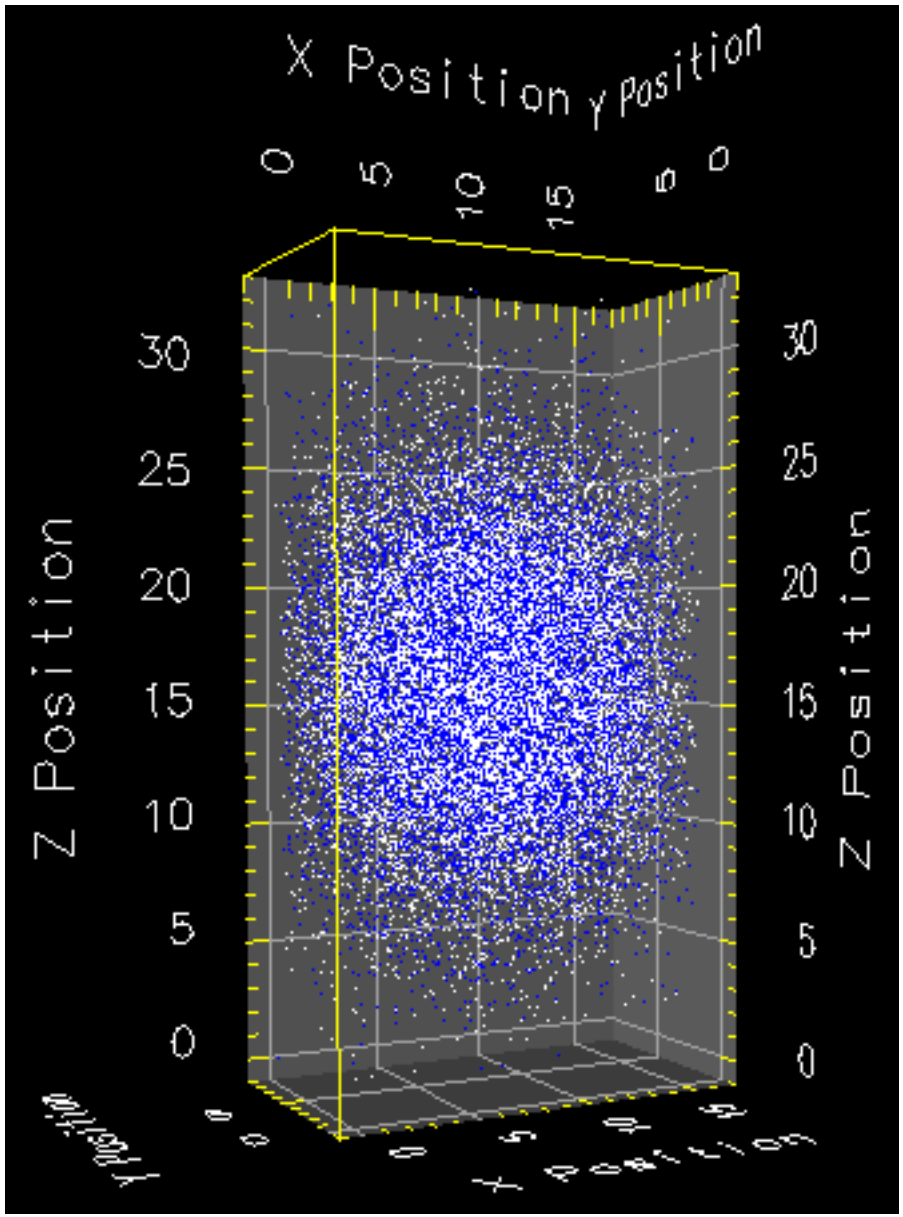
```
PROGRAM beps3k
 use partition_class; use plasma_class
 type (distf) :: backdf, beamdf        ! declare all objects...
 call MPI_INIT(ierror)
 call species_init(electrons, qme, qbme, np)    ! initialize
 call fields_init(cdensity, nx, ny, nz)         ! objects...
 call fields_init(efield, nx, ny, nz)
 DO itime = 1,500
    call fields_solve(cdensity, efield)
    call plasma_getpe(energ, efield)
    call plasma_push(electrons, efield, edges, dt)
    call plasma_pmove(electrons, edges)
    call plasma_dpost(electrons, cdensity, edges)
 END DO
 call fields_destroy(efield) ; call fields_destroy(cdensity)
 call species_destroy(electrons)  ! destroy dynamic objects
 call MPI_FINALIZE(ierror)
END PROGRAM beps3k
```

Object-Oriented Fortran 90 Parallel PIC Model



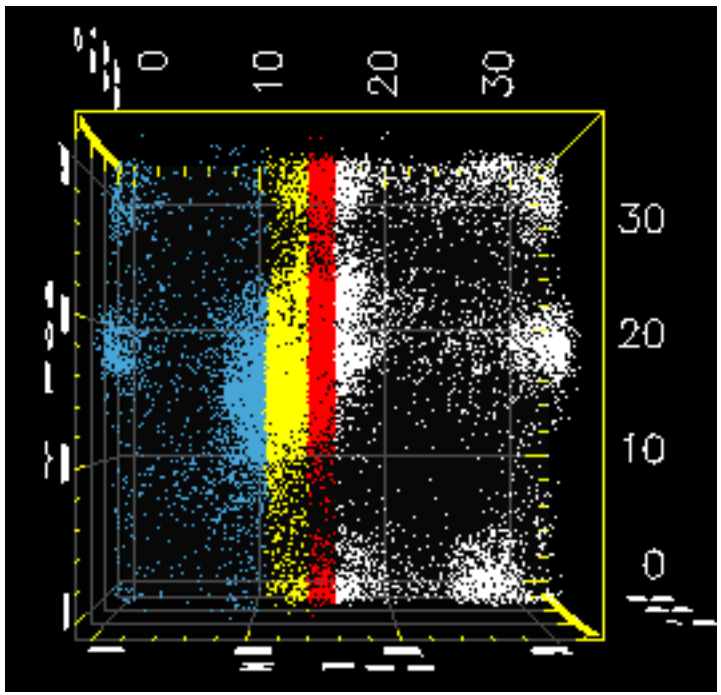
OO techniques capture essential abstractions (particles, fields, relationships). This enhances collaborations and promotes safer software design.

3D Free-expansion and Gravitational Experiments



Ion-Electron expansion from laser-fusion experiment.

Partitioning from dynamic load balancing in gravitational problem.

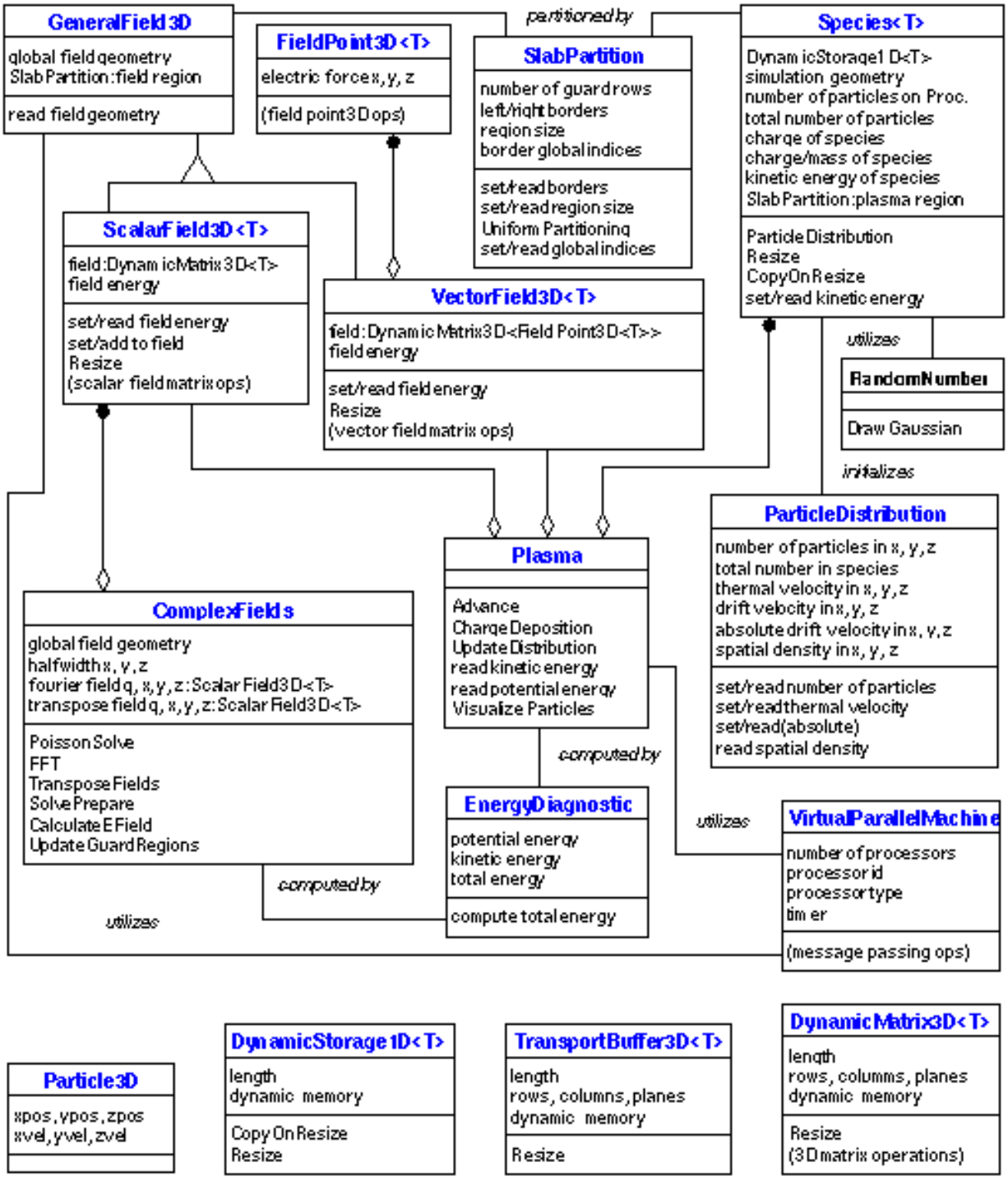


"Sketch" Of Object-Oriented C++ PIC Code

```
void main() {
 VPMachine vpm;                    ! declare and init objects...
 DistFunction backdf, beamdf;
 Species< Particle3D > electrons;
 ScalarField3D< float > cdensity( nx, ny, nz, vpm );
 VectorField3D< float > efield( nx, ny, nz, vpm );
 Plasma plasma;
 Fields3D fields( inx, iny, inz, vpm );
 for ( int i=0; i < N_STEPS; i++ ) {
    fields.Solve(cdensity, efield, vpm);
    plasma.getpe(efield, energy);
    plasma.push(electrons, efield, dt);
    plasma.pmove(electrons, vpm);
    plasma.dpost(electrons, cdensity);
 }
 vpm.ParFinal();
} // End Main...
```

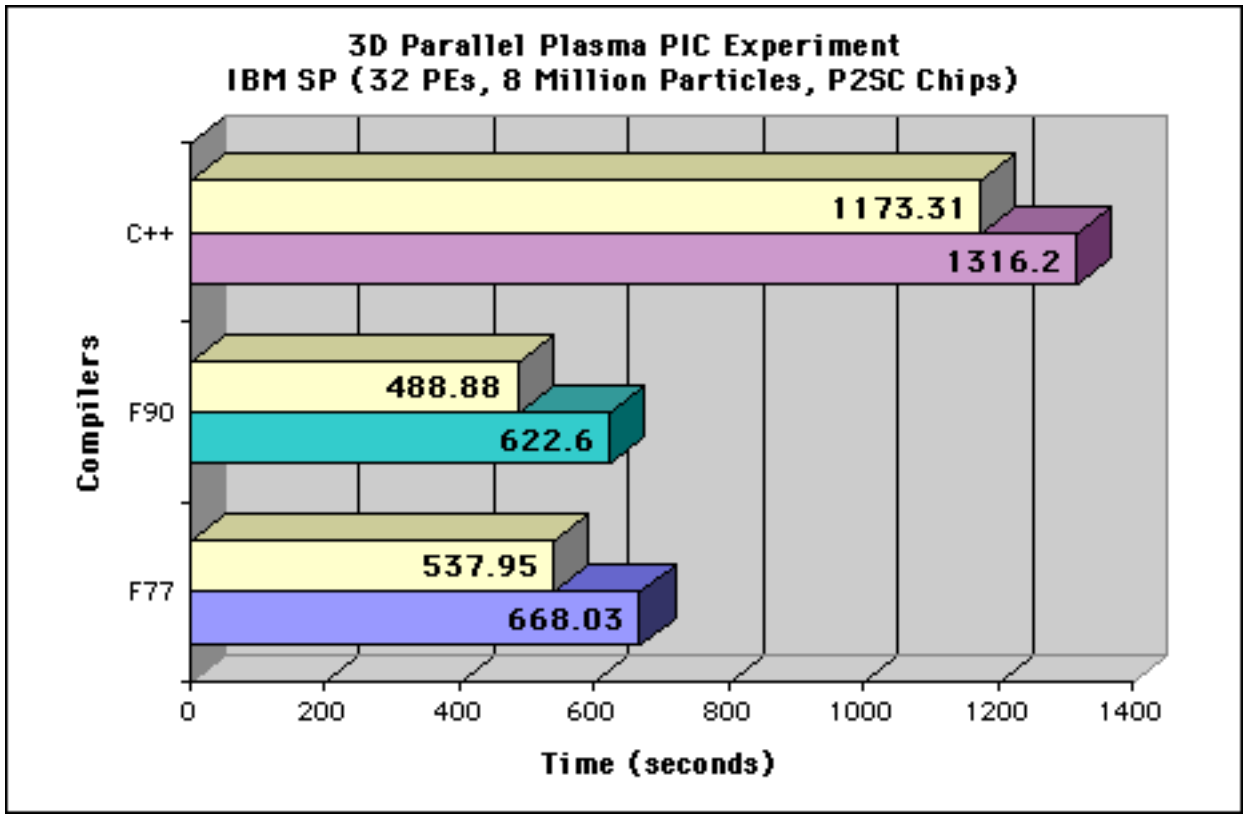
Earth and Space Sciences Project
(625-20)

C++ Parallel Object Oriented Model



JPL

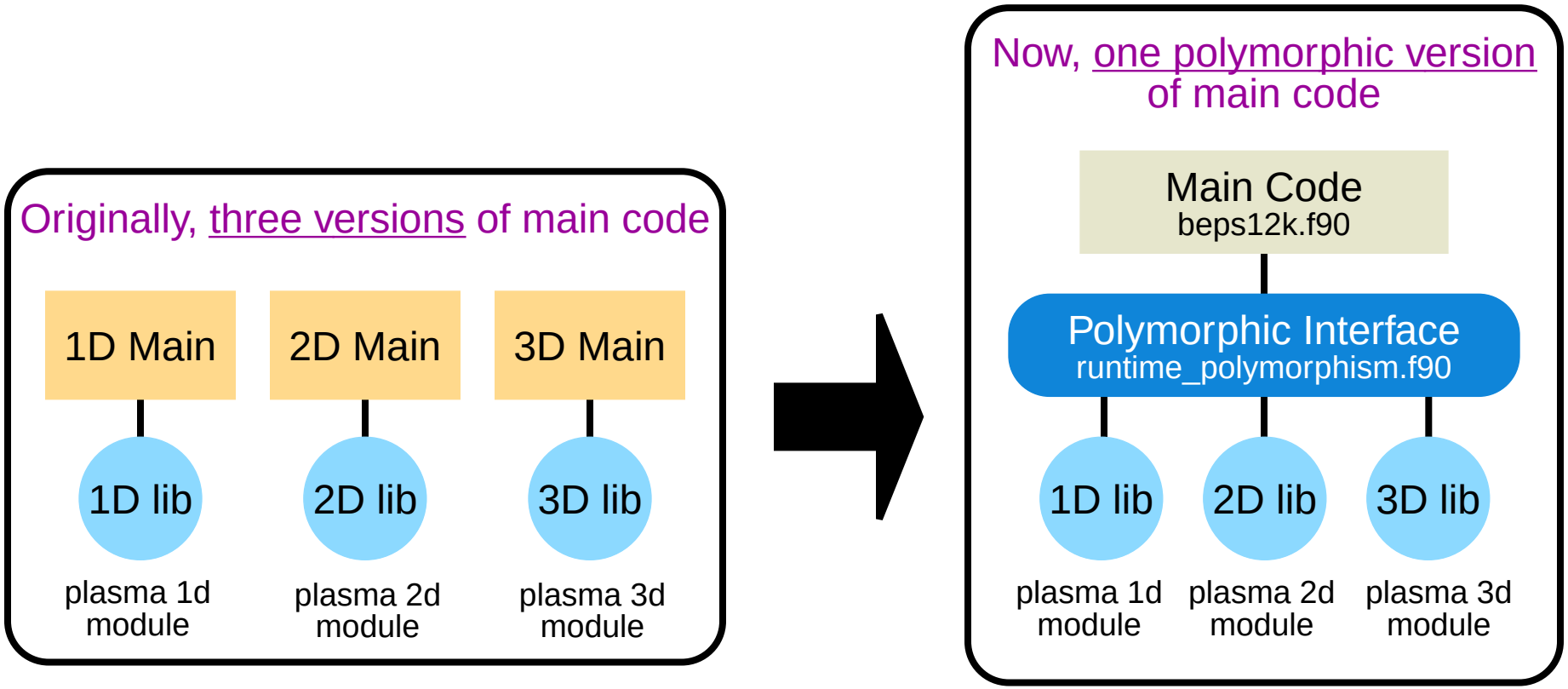
IBM F77, IBM F90, and KAI C++ Parallel Performance Comparison on CTC IBM SP2 with MPI



KAI C++ Compilation (2hours) IBM F90 Compilation (5 minutes)

Object-Oriented approaches applied to Fortran 90 outperform Fortran 77 and C++ in this application. (Results in yellow use P2SC Super Chip hardware optimizations.)

Combine Existing Modules in a more Powerful Way with a Polymorphic Interface



Fortran 90 Supports Abstract Data Types

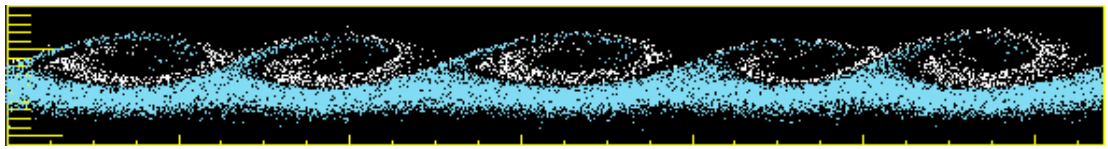
- Simplifies program interfaces and hides implementation details
- Helps build codes from modular pieces written by different authors
  - Enhances software collaboration
  - Increases program safety
  - Older codes are safer and easier to use
  - Allows code changes without impacting users
  - Encapsulates old codes, advancing them to new standards
  - Program can be expressed in a more natural way

```
PROGRAM p_amr
USE mpi_module
USE mesh_module
type (mesh) :: pmesh
  call create(pmesh,'file')
  call partition(pmesh)
  call refinement(pmesh)
  call migration(pmesh)
  call visualize(pmesh)
END PROGRAM p_amr
```

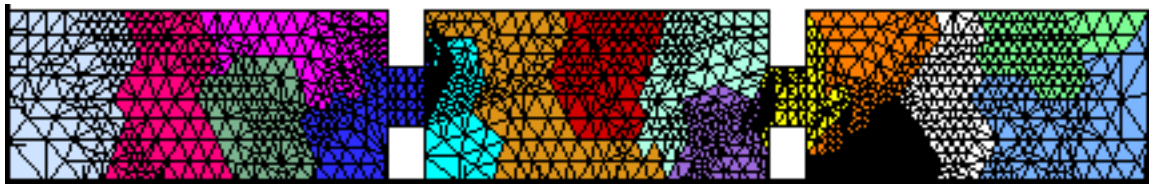
Complex Data Structures Are Now Possible

- Makes more realistic, complex, & dynamic simulations more routine (AMR, Multigrid, Plasma Modeling, ...)

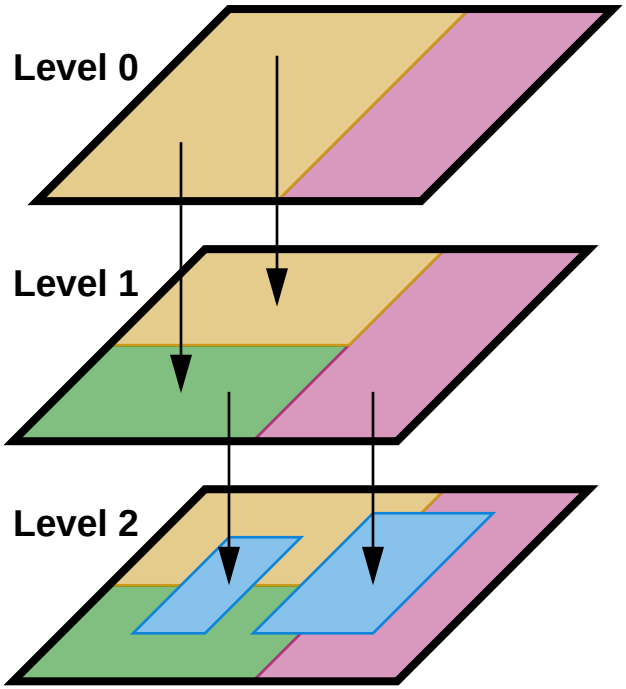
Beam-Plasma Instability Experiment



Adaptive Mesh Refinement



Multigrid Numerical Solvers



HPCC/Earth and Space Sciences Project

PYRAMID: Parallel Unstructured Adaptive Mesh Refinement

PYRAMID

Modern... Simple... Efficient... Scalable...

Technology Description

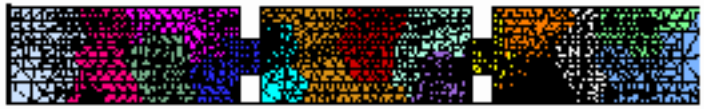
An advanced software library supporting parallel adaptive mesh refinementin large-scale, adaptive scientific & engineering simulations.

State-of-the-Art Design

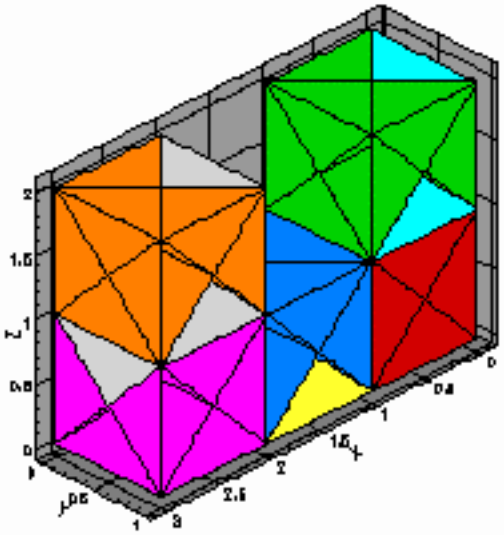
- Efficient object-oriented design in Fortran 90 and MPI
- Automatic mesh quality control & dynamic load balancing
- Scalableto hundreds of processors & millions of elements

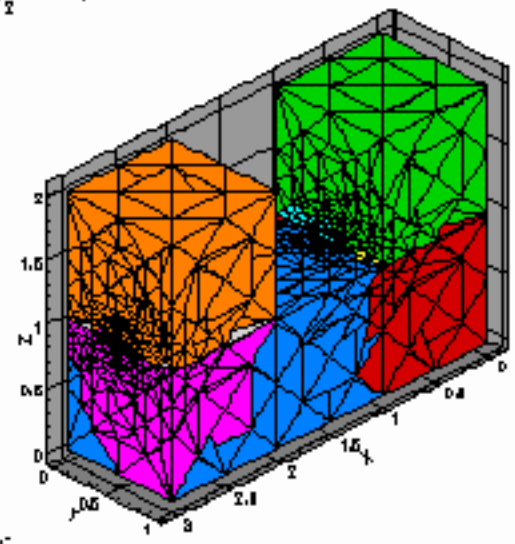
Application Arena

- Computer Modeling & Simulation Applications with complex geometry
- Electromagnetic and semiconductor device modeling
- Structural/Mechanical/Fluid dynamics applications



2D Adaptive Refinement on Waveguide Filter





3D Successive Parallel Adaptive Refinements on Cray T3E

Accomplishments

- 3D library developed and tested on GSFC Cray T3E
- System applied to a variety of meshes
- Publications and presentations

John Z. Lou, Charles D. Norton, & Thomas A. Cwik
High Performance Computing Systems and Applications Group
http://www-hpc.jpl.nasa.gov/APPS/AMR

Data Structures and AMR Relationships

Fortran 90 Structures (NASA JPL)

OO Approach : Use OO aspects of Fortran 90 to organize mesh structure

```
module mesh_module                          ! abstract of module...
use mpi_module ; use heapsort_module ; private
    public :: jpl_mesh_repartition, ...
    type mesh
      type(node), dimension(:), pointer :: nodes
      type(edge), dimension(:), pointer :: edges
      type(element), dimension(:), pointer :: elements
      type(b_element), dimension(:), pointer :: bndy_elements
    end type mesh
contains
    ! module member routines...
end module mesh_module
```

Data Structures and AMR Relationships

Fortran 90 Structures (NASA JPL)

Using Abstractions: Initialization Section

```
program pamr
use mesh_module
     implicit none
     ! statments omitted...
        type(mesh) :: in_mesh, parent_mesh
        call MPI_INIT(ierror)
        call jpl_mesh_create_incore(in_mesh, in_file)
        call jpl_mesh_repartition(in_mesh)
          ! adaptive refinement loop...
        call jpl_mesh_visualize(in_mesh, “visfile.plt”)
        call MPI_FINALIZE(ierror)
end program pamr
```

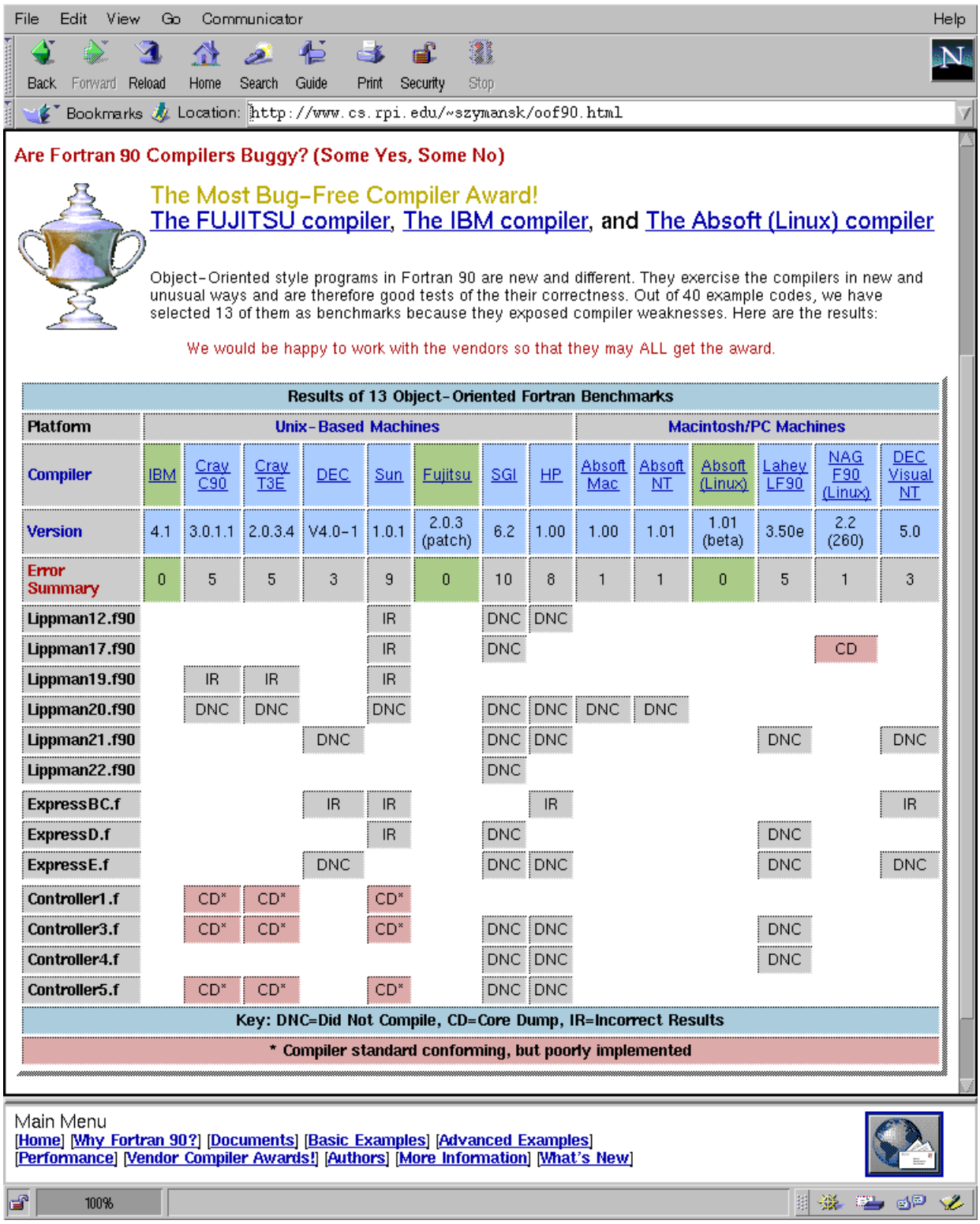
Data Structures and AMR Relationships

Fortran 90 Structures (NASA JPL)

Using Abstractions: Loop Section

```
program pamr
! adaptive refinement loop...
    do i = 1, refine_level
      call jpl_mesh_error_est(in_mesh, parent_mesh)
      call jpl_mesh_logical_amr(in_mesh)
      call jpl_mesh_repartition(in_mesh)
      call jpl_mesh_physical_amr(in_mesh)
    end do
end program pamr
```

- Error estimation is not part of the library.



File Edit View Go Communicator Help

Back Forward Reload Home Search Guide Print Security Stop

Bookmarks Location: http://www.cs.rpi.edu/~szymansk/oof90.html

Are Fortran 90 Compilers Buggy? (Some Yes, Some No)

The Most Bug-Free Compiler Award!

The FUJITSU compiler, The IBM compiler, and The Absoft (Linux) compiler

Object-Oriented style programs in Fortran 90 are new and different. They exercise the compilers in new and unusual ways and are therefore good tests of the their correctness. Out of 40 example codes, we have selected 13 of them as benchmarks because they exposed compiler weaknesses. Here are the results:

We would be happy to work with the vendors so that they may ALL get the award.

Results of 13 Object-Oriented Fortran Benchmarks

Platform	Unix-Based Machines								Macintosh/PC Machines					
Compiler	IBM	Cray C90	Cray T3E	DEC	Sun	Fujitsu	SGI	HP	Absoft Mac	Absoft NT	Absoft (Linux)	Lahey LF90	NAG F90 (Linux)	DEC Visual NT
Version	4.1	3.0.1.1	2.0.3.4	V4.0-1	1.0.1	2.0.3 (patch)	6.2	1.00	1.00	1.01	1.01 (beta)	3.50e	2.2 (260)	5.0
Error Summary	0	5	5	3	9	0	10	8	1	1	0	5	1	3
Lippman12.f90					IR		DNC	DNC						
Lippman17.f90					IR		DNC						CD	
Lippman19.f90		IR	IR		IR									
Lippman20.f90		DNC	DNC		DNC		DNC	DNC	DNC	DNC				
Lippman21.f90				DNC			DNC	DNC				DNC		DNC
Lippman22.f90							DNC							
ExpressBC.f				IR	IR			IR						IR
ExpressD.f					IR		DNC					DNC		
ExpressE.f				DNC			DNC	DNC				DNC		DNC
Controller1.f		CD*	CD*		CD*									
Controller3.f		CD*	CD*		CD*		DNC	DNC				DNC		
Controller4.f							DNC	DNC				DNC		
Controller5.f		CD*	CD*		CD*		DNC	DNC						

Key: DNC=Did Not Compile, CD=Core Dump, IR=Incorrect Results

* Compiler standard conforming, but poorly implemented

Main Menu

[Home] [Why Fortran 90?] [Documents] [Basic Examples] [Advanced Examples]
[Performance] [Vendor Compiler Awards!] [Authors] [More Information] [What's New]

100%

JPL

Some Features Approved for Fortran 2000

Firm Requirements (developed by J3)

- Derived Type I/O, Procedure Pointers, Internationalization, Interoperability with C,...
- OO Features: Inheritance, Polymorphism, Parameterized Derived Types, Constructors/Destructors

Minor Technical Enhancements (optional, time permitting)

- Increased Statement Length, Intent for Pointer Arguments, Command Line Arguments and Variables, Volatile Attribute,...

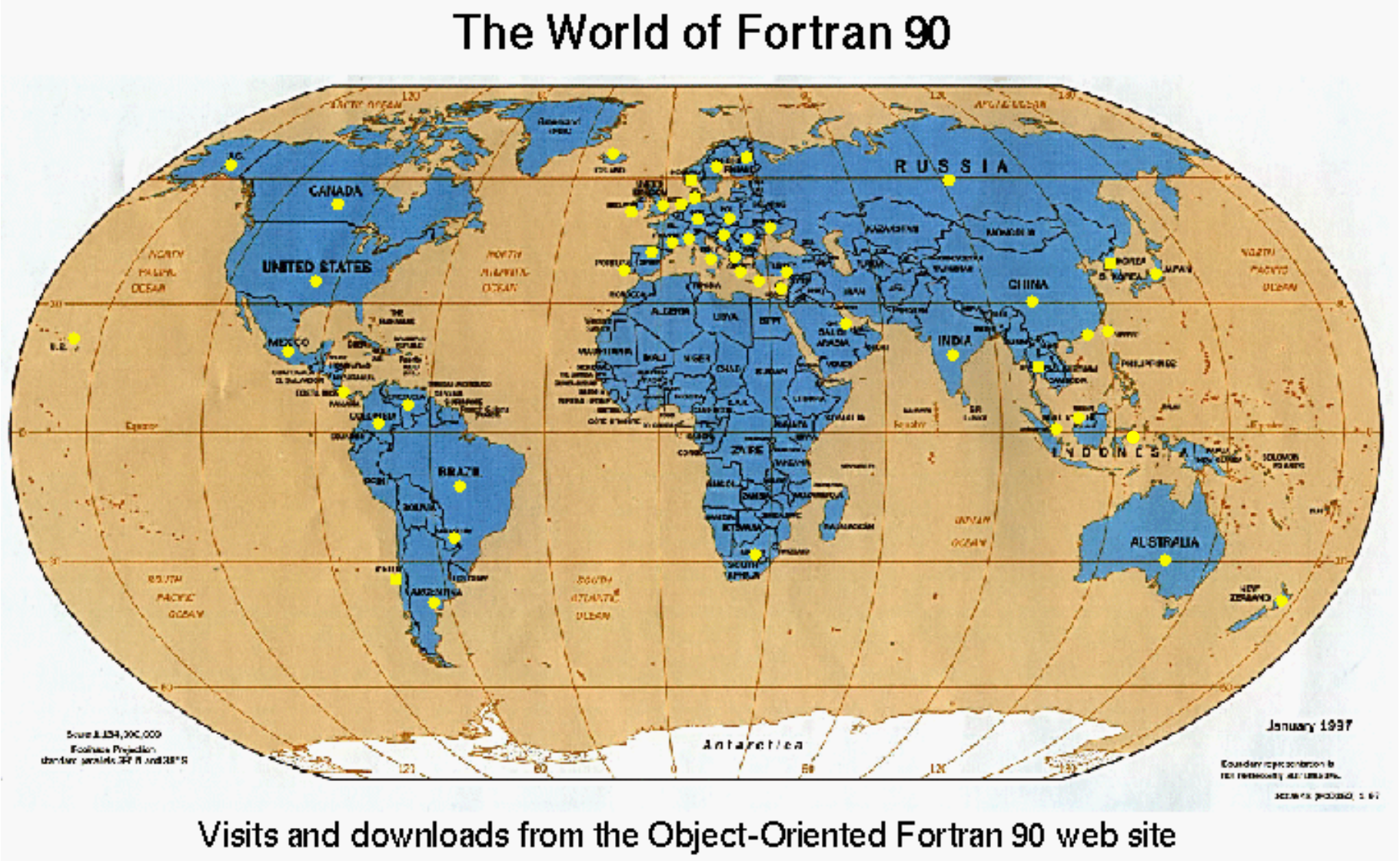
Minor Technical Enhancements (lowest priority)

- Public and Private derived type components,...

Conclusions

Fortran 90 is useful for complex, scientific parallel programming

- Safer and faster development than C++
- Inheritance/Polymorphism is not supported, but can be emulated (included in Fortran 2000)
- Performance is equalvalent, or exceeds, Fortran 77/C++
- Separation of inheritance and polymorphism is important when codes have similar interfaces, but algorithms are not related
- Inheritance, rarely used, typically appeared in the non-scientific code sections


The World of Fortran 90
CANADA
UNITED STATES
MEXICO
BRAZIL
ARGENTINA
RUSSIA
CHINA
INDIA
INDONESIA
AUSTRALIA
Antarctica
January 1997
Visits and downloads from the Object-Oriented Fortran 90 web site